In [24]:
```
## Artificial Intelligence Project
## Ansh Gupta
## 19BIT0220
## How Artificial Intelligence helps to combat Cyber Bullying

import re
import os

import pandas as pd
import numpy as np
import string
from collections import Counter
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras import losses
from tensorflow.keras import regularizers
from tensorflow.keras import preprocessing
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
## Importing the files by collabrating with google to take the access for Kag
from google.colab import files
files.upload()
```

In [4]:
```
## Installation of Kaggle in my Jupiter Library
!pip install -q kaggle
```

In [5]:
```
!mkdir -p ~/.kaggle
```

In [6]:
```
!cp kaggle.json ~/.kaggle/
```

In [7]:
```
!chmod 600 /root/.kaggle/kaggle.json
```

In [9]:
```
# Fetching the dataset of tweets and download it in our library
!kaggle datasets download -d vkrahul/twitter-hate-speech
```

```
Downloading twitter-hate-speech.zip to /content
  0% 0.00/1.89M [00:00<?, ?B/s]
100% 1.89M/1.89M [00:00<00:00, 62.3MB/s]
```

In [10]:
```
# Downloaded in zip file , so now unzipping the file
!unzip /content/twitter-hate-speech.zip
```

```
Archive:  /content/twitter-hate-speech.zip
  inflating: test_tweets_anuFYb8.csv
  inflating: train_E6oV3lV.csv
```

In [12]:

```
# Reading the raw data set
raw_data = pd.read_csv('/content/train_tweets.csv')
data = raw_data.copy()
data.drop(columns=['id'], axis=1, inplace=True)
data.head()
```

Out[12]:

	label	tweet
0	0	@user when a father is dysfunctional and is s...
1	0	@user @user thanks for #lyft credit i can't us...
2	0	bihday your majesty
3	0	#model i love u take with u all the time in ...
4	0	factsguide: society now #motivation

In [17]:

```
# Through this we found that how much is normal tweets and how much are of no

print(np.round(data['label'].value_counts()[0]/len(data) * 100, 2), "% are No
print(np.round(data['label'].value_counts()[1]/len(data) * 100, 2), "% are Ha
```

```
92.99 % are Normal Speech
7.01 % are Hate Speech
```

In [22]:

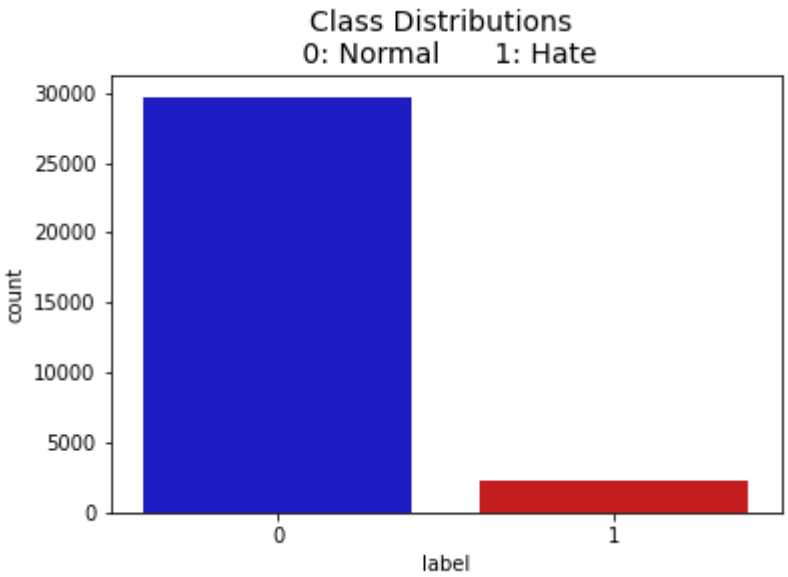
```
# Plotting it into the graphs

colors = ["#0101DF", "#DF0101"]

sns.countplot('label', data=data, palette=colors)
plt.title('Class Distributions \n 0: Normal      1: Hate', fontsize=14)
```

Out[22]:

```
Text(0.5, 1.0, 'Class Distributions \n 0: Normal      1: Hate')
```



In [32]:

```
# Till now we have seen only for texts but emojis are also coming into bullyi
# So for that we are now filtering the emojis

def remove_emoji(text):
```

```
    emoji_pattern = re.compile("["
                u"\U0001F600-\U0001F64F" #emoticons
                u"\U0001F300-\U0001F5FF" #symbols & pictograms
                u"\U0001F680-\U0001F6FF" #transport & map symbols
                u"\U0001F1E0-\U0001F1FF" #flags(ios)
                u"\U00002702-\U000027B0"
                u"\U000024C2-\U0001F251"
                "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)

def clean_text(text):
    delete_dict = {sp_character: '' for sp_character in string.punctuation}
    delete_dict[' '] = ' '
    table = str.maketrans(delete_dict)
    text1 = text.translate(table)
    textArr = text1.split()
    text2 = ' '.join([w for w in textArr if(not w.isdigit() and (not w.isdigi

    return text2.lower()
```

In [35]:

```
# now we are removing emoji and removing test from our library
smptw = '@user #white #supremacists want everyone to see the new â€¦ #birdsâ€
smptw = remove_emoji(smptw)
smptw = clean_text(smptw)
print(smptw)
```

```
user white supremacists want everyone birdsâ€¦ movie hereâ€¦ s
```

In [36]:

```
data['tweet'] = data['tweet'].apply(remove_emoji)
data['tweet'] = data['tweet'].apply(clean_text)
data['num_words_text'] = data['tweet'].apply(lambda x : len(str(x).split()))

train_data, val_data = train_test_split(data, test_size=0.2)
train_data.reset_index(drop=True, inplace=True)
val_data.reset_index(drop=True, inplace=True)
```

In [38]:

```
# Now we are training the data set to count the words
test_data = val_data
print("==== Train Data ====")
print(train_data['label'].value_counts())
print(len(train_data))

print("==== Test Data ====")
print(test_data['label'].value_counts())
print(len(test_data))
```

```
==== Train Data ====
0    23784
1     1785
Name: label, dtype: int64
25569
==== Test Data ====
0    5936
1     457
Name: label, dtype: int64
6393
```

In [39]:

```
X_train, X_valid, y_train, y_valid = train_test_split(train_data['tweet'].tol
print("Train Data len: ", len(X_train))
print("Class distribution: ", Counter(y_train))
```

```
print("Validation Data len: ", len(X_valid))
print("Class distribution: ", Counter(y_valid))
```

```
Train Data len:  20455
Class distribution:  Counter({0: 19027, 1: 1428})
Validation Data len:  5114
Class distribution:  Counter({0: 4757, 1: 357})
```

In [42]:

```
X_train[5]
```

Out[42]:

```
'user couple knew thatll together forever just broke upð\x9f\x92\x94ð\x9f\x98
\x9eð\x9f\x98\x94 dilmer'
```

In [43]:

```
num_words=50000

tokenizer = Tokenizer(num_words=num_words, oov_token="<UNK>")
tokenizer.fit_on_texts(X_train)
```

In [45]:

```
# Now we are making an array to do the filteration

x_train = np.array(tokenizer.texts_to_sequences(X_train))
x_valid = np.array(tokenizer.texts_to_sequences(X_valid))
x_test = np.array(tokenizer.texts_to_sequences(test_data['tweet'].tolist()))

maxlen=50
x_train = pad_sequences(x_train, padding='post', maxlen=maxlen)
x_valid = pad_sequences(x_valid, padding='post', maxlen=maxlen)
x_test = pad_sequences(x_test, padding='post', maxlen=maxlen)

train_labels = np.asarray(y_train)
valid_labels = np.asarray(y_valid)
test_labels = np.asarray(test_data['label'].tolist())

print("Train data: ", len(x_train))
print("Validation data: ", len(x_valid))
print("Test data: ", len(x_test))

#Tensorflow dataset
train_ds = tf.data.Dataset.from_tensor_slices((x_train, train_labels))
valid_ds = tf.data.Dataset.from_tensor_slices((x_valid, valid_labels))
test_ds = tf.data.Dataset.from_tensor_slices((x_test, test_labels))
```

```
Train data:  20455
Validation data:  5114
Test data:  6393
```

In [47]:

```
max_features = 50000
embedding_dim = 16
sequence_length=maxlen

model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(max_features + 1, embedding_dim, input_le
model.add(tf.keras.layers.Dropout(0.4))
model.add(tf.keras.layers.LSTM(embedding_dim, dropout=0.2, recurrent_dropout=
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(512, activation='relu', kernel_regularizer=re
model.add(tf.keras.layers.Dropout(0.4))
model.add(tf.keras.layers.Dense(8, activation='relu', kernel_regularizer=regu
model.add(tf.keras.layers.Dropout(0.4))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 50, 16)            800016
_________________________________________________________________
dropout (Dropout)            (None, 50, 16)            0
_________________________________________________________________
lstm (LSTM)                  (None, 50, 16)            2112
_________________________________________________________________
flatten (Flatten)            (None, 800)               0
_________________________________________________________________
dense (Dense)                (None, 512)               410112
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 4104
_________________________________________________________________
dropout_2 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 9
=================================================================
Total params: 1,216,353
Trainable params: 1,216,353
Non-trainable params: 0
_________________________________________________________________
```

In [48]:
```
model.compile(loss=tf.keras.losses.BinaryCrossentropy(), optimizer=tf.keras.o
```

In [49]:
```
epochs=100
history=model.fit(train_ds.shuffle(5000).batch(1024), epochs=epochs, validati
```

```
Epoch 1/100
20/20 [==============================] - 10s 305ms/step - loss: 3.8426 - binar
y_accuracy: 0.8469 - val_loss: 1.5764 - val_binary_accuracy: 0.9302
Epoch 2/100
20/20 [==============================] - 6s 287ms/step - loss: 1.3267 - binary
_accuracy: 0.9293 - val_loss: 0.6721 - val_binary_accuracy: 0.9302
Epoch 3/100
20/20 [==============================] - 6s 278ms/step - loss: 0.6547 - binary
_accuracy: 0.9300 - val_loss: 0.4740 - val_binary_accuracy: 0.9302
Epoch 4/100
20/20 [==============================] - 5s 273ms/step - loss: 0.4983 - binary
_accuracy: 0.9303 - val_loss: 0.4181 - val_binary_accuracy: 0.9302
Epoch 5/100
20/20 [==============================] - 6s 276ms/step - loss: 0.4537 - binary
_accuracy: 0.9309 - val_loss: 0.3912 - val_binary_accuracy: 0.9302
Epoch 6/100
20/20 [==============================] - 6s 281ms/step - loss: 0.4314 - binary
_accuracy: 0.9293 - val_loss: 0.3725 - val_binary_accuracy: 0.9302
Epoch 7/100
20/20 [==============================] - 6s 280ms/step - loss: 0.4094 - binary
_accuracy: 0.9307 - val_loss: 0.3572 - val_binary_accuracy: 0.9302
Epoch 8/100
20/20 [==============================] - 6s 278ms/step - loss: 0.3968 - binary
_accuracy: 0.9299 - val_loss: 0.3455 - val_binary_accuracy: 0.9302
Epoch 9/100
20/20 [==============================] - 6s 276ms/step - loss: 0.3771 - binary
_accuracy: 0.9308 - val_loss: 0.3337 - val_binary_accuracy: 0.9302
```

```
Epoch 10/100
20/20 [==============================] - 6s 278ms/step - loss: 0.3647 - binary
_accuracy: 0.9295 - val_loss: 0.3186 - val_binary_accuracy: 0.9302
Epoch 11/100
20/20 [==============================] - 6s 287ms/step - loss: 0.3440 - binary
_accuracy: 0.9305 - val_loss: 0.2948 - val_binary_accuracy: 0.9302
Epoch 12/100
20/20 [==============================] - 5s 268ms/step - loss: 0.3189 - binary
_accuracy: 0.9335 - val_loss: 0.2762 - val_binary_accuracy: 0.9488
Epoch 13/100
20/20 [==============================] - 5s 265ms/step - loss: 0.2949 - binary
_accuracy: 0.9419 - val_loss: 0.3027 - val_binary_accuracy: 0.9515
Epoch 14/100
20/20 [==============================] - 6s 288ms/step - loss: 0.2840 - binary
_accuracy: 0.9474 - val_loss: 0.2681 - val_binary_accuracy: 0.9499
Epoch 15/100
20/20 [==============================] - 5s 269ms/step - loss: 0.2529 - binary
_accuracy: 0.9530 - val_loss: 0.2733 - val_binary_accuracy: 0.9574
Epoch 16/100
20/20 [==============================] - 6s 285ms/step - loss: 0.2257 - binary
_accuracy: 0.9573 - val_loss: 0.2517 - val_binary_accuracy: 0.9609
Epoch 17/100
20/20 [==============================] - 6s 287ms/step - loss: 0.2201 - binary
_accuracy: 0.9572 - val_loss: 0.2635 - val_binary_accuracy: 0.9585
Epoch 18/100
20/20 [==============================] - 6s 276ms/step - loss: 0.2123 - binary
_accuracy: 0.9576 - val_loss: 0.2580 - val_binary_accuracy: 0.9507
Epoch 19/100
20/20 [==============================] - 6s 280ms/step - loss: 0.2107 - binary
_accuracy: 0.9594 - val_loss: 0.2620 - val_binary_accuracy: 0.9568
Epoch 20/100
20/20 [==============================] - 6s 292ms/step - loss: 0.2008 - binary
_accuracy: 0.9570 - val_loss: 0.3010 - val_binary_accuracy: 0.9601
Epoch 21/100
20/20 [==============================] - 6s 282ms/step - loss: 0.1954 - binary
_accuracy: 0.9610 - val_loss: 0.2924 - val_binary_accuracy: 0.9595
Epoch 22/100
20/20 [==============================] - 6s 288ms/step - loss: 0.1861 - binary
_accuracy: 0.9655 - val_loss: 0.2735 - val_binary_accuracy: 0.9603
Epoch 23/100
20/20 [==============================] - 6s 277ms/step - loss: 0.1864 - binary
_accuracy: 0.9620 - val_loss: 0.2524 - val_binary_accuracy: 0.9634
Epoch 24/100
20/20 [==============================] - 6s 282ms/step - loss: 0.1760 - binary
_accuracy: 0.9634 - val_loss: 0.2867 - val_binary_accuracy: 0.9589
Epoch 25/100
20/20 [==============================] - 6s 278ms/step - loss: 0.1759 - binary
_accuracy: 0.9642 - val_loss: 0.2643 - val_binary_accuracy: 0.9613
Epoch 26/100
20/20 [==============================] - 5s 275ms/step - loss: 0.1751 - binary
_accuracy: 0.9622 - val_loss: 0.3216 - val_binary_accuracy: 0.9619
Epoch 27/100
20/20 [==============================] - 6s 278ms/step - loss: 0.1725 - binary
_accuracy: 0.9649 - val_loss: 0.3355 - val_binary_accuracy: 0.9611
Epoch 28/100
20/20 [==============================] - 6s 280ms/step - loss: 0.1750 - binary
_accuracy: 0.9623 - val_loss: 0.2853 - val_binary_accuracy: 0.9615
Epoch 29/100
20/20 [==============================] - 6s 286ms/step - loss: 0.1797 - binary
_accuracy: 0.9612 - val_loss: 0.2911 - val_binary_accuracy: 0.9603
Epoch 30/100
20/20 [==============================] - 6s 276ms/step - loss: 0.1654 - binary
_accuracy: 0.9667 - val_loss: 0.2530 - val_binary_accuracy: 0.9447
Epoch 31/100
```

```
20/20 [==============================] - 6s 281ms/step - loss: 0.1636 - binary
_accuracy: 0.9658 - val_loss: 0.2466 - val_binary_accuracy: 0.9611
Epoch 32/100
20/20 [==============================] - 6s 286ms/step - loss: 0.1668 - binary
_accuracy: 0.9649 - val_loss: 0.2722 - val_binary_accuracy: 0.9582
Epoch 33/100
20/20 [==============================] - 5s 267ms/step - loss: 0.1659 - binary
_accuracy: 0.9632 - val_loss: 0.2380 - val_binary_accuracy: 0.9533
Epoch 34/100
20/20 [==============================] - 6s 288ms/step - loss: 0.1728 - binary
_accuracy: 0.9587 - val_loss: 0.2523 - val_binary_accuracy: 0.9617
Epoch 35/100
20/20 [==============================] - 6s 282ms/step - loss: 0.1602 - binary
_accuracy: 0.9640 - val_loss: 0.3262 - val_binary_accuracy: 0.9619
Epoch 36/100
20/20 [==============================] - 5s 269ms/step - loss: 0.1632 - binary
_accuracy: 0.9641 - val_loss: 0.2564 - val_binary_accuracy: 0.9603
Epoch 37/100
20/20 [==============================] - 6s 278ms/step - loss: 0.1588 - binary
_accuracy: 0.9649 - val_loss: 0.2418 - val_binary_accuracy: 0.9562
Epoch 38/100
20/20 [==============================] - 6s 290ms/step - loss: 0.1517 - binary
_accuracy: 0.9654 - val_loss: 0.2769 - val_binary_accuracy: 0.9615
Epoch 39/100
20/20 [==============================] - 6s 282ms/step - loss: 0.1525 - binary
_accuracy: 0.9641 - val_loss: 0.2817 - val_binary_accuracy: 0.9619
Epoch 40/100
20/20 [==============================] - 6s 284ms/step - loss: 0.1480 - binary
_accuracy: 0.9650 - val_loss: 0.2192 - val_binary_accuracy: 0.9597
Epoch 41/100
20/20 [==============================] - 6s 283ms/step - loss: 0.1463 - binary
_accuracy: 0.9672 - val_loss: 0.2385 - val_binary_accuracy: 0.9554
Epoch 42/100
20/20 [==============================] - 6s 283ms/step - loss: 0.1472 - binary
_accuracy: 0.9635 - val_loss: 0.2490 - val_binary_accuracy: 0.9583
Epoch 43/100
20/20 [==============================] - 6s 288ms/step - loss: 0.1462 - binary
_accuracy: 0.9616 - val_loss: 0.2588 - val_binary_accuracy: 0.9593
Epoch 44/100
20/20 [==============================] - 6s 290ms/step - loss: 0.1449 - binary
_accuracy: 0.9661 - val_loss: 0.2577 - val_binary_accuracy: 0.9597
Epoch 45/100
20/20 [==============================] - 6s 278ms/step - loss: 0.1446 - binary
_accuracy: 0.9647 - val_loss: 0.2576 - val_binary_accuracy: 0.9593
Epoch 46/100
20/20 [==============================] - 6s 285ms/step - loss: 0.1422 - binary
_accuracy: 0.9648 - val_loss: 0.2633 - val_binary_accuracy: 0.9605
Epoch 47/100
20/20 [==============================] - 5s 273ms/step - loss: 0.1405 - binary
_accuracy: 0.9634 - val_loss: 0.2114 - val_binary_accuracy: 0.9564
Epoch 48/100
20/20 [==============================] - 5s 274ms/step - loss: 0.1450 - binary
_accuracy: 0.9642 - val_loss: 0.2516 - val_binary_accuracy: 0.9599
Epoch 49/100
20/20 [==============================] - 6s 285ms/step - loss: 0.1413 - binary
_accuracy: 0.9642 - val_loss: 0.2751 - val_binary_accuracy: 0.9601
Epoch 50/100
20/20 [==============================] - 6s 282ms/step - loss: 0.1412 - binary
_accuracy: 0.9646 - val_loss: 0.2693 - val_binary_accuracy: 0.9583
Epoch 51/100
20/20 [==============================] - 6s 285ms/step - loss: 0.1350 - binary
_accuracy: 0.9682 - val_loss: 0.2421 - val_binary_accuracy: 0.9601
Epoch 52/100
20/20 [==============================] - 6s 288ms/step - loss: 0.1375 - binary
```

```
_accuracy: 0.9635 - val_loss: 0.2507 - val_binary_accuracy: 0.9599
Epoch 53/100
20/20 [==============================] - 6s 278ms/step - loss: 0.1369 - binary
_accuracy: 0.9662 - val_loss: 0.2391 - val_binary_accuracy: 0.9572
Epoch 54/100
20/20 [==============================] - 6s 277ms/step - loss: 0.1331 - binary
_accuracy: 0.9655 - val_loss: 0.2599 - val_binary_accuracy: 0.9580
Epoch 55/100
20/20 [==============================] - 6s 286ms/step - loss: 0.1318 - binary
_accuracy: 0.9635 - val_loss: 0.2644 - val_binary_accuracy: 0.9603
Epoch 56/100
20/20 [==============================] - 6s 280ms/step - loss: 0.1345 - binary
_accuracy: 0.9643 - val_loss: 0.2852 - val_binary_accuracy: 0.9607
Epoch 57/100
20/20 [==============================] - 6s 286ms/step - loss: 0.1452 - binary
_accuracy: 0.9626 - val_loss: 0.2484 - val_binary_accuracy: 0.9625
Epoch 58/100
20/20 [==============================] - 6s 281ms/step - loss: 0.1424 - binary
_accuracy: 0.9622 - val_loss: 0.2469 - val_binary_accuracy: 0.9523
Epoch 59/100
20/20 [==============================] - 5s 273ms/step - loss: 0.1396 - binary
_accuracy: 0.9640 - val_loss: 0.2354 - val_binary_accuracy: 0.9576
Epoch 60/100
20/20 [==============================] - 6s 280ms/step - loss: 0.1389 - binary
_accuracy: 0.9630 - val_loss: 0.2554 - val_binary_accuracy: 0.9613
Epoch 61/100
20/20 [==============================] - 6s 289ms/step - loss: 0.1297 - binary
_accuracy: 0.9652 - val_loss: 0.2826 - val_binary_accuracy: 0.9619
Epoch 62/100
20/20 [==============================] - 6s 285ms/step - loss: 0.1443 - binary
_accuracy: 0.9632 - val_loss: 0.2067 - val_binary_accuracy: 0.9574
Epoch 63/100
20/20 [==============================] - 5s 274ms/step - loss: 0.1502 - binary
_accuracy: 0.9612 - val_loss: 0.2760 - val_binary_accuracy: 0.9625
Epoch 64/100
20/20 [==============================] - 6s 283ms/step - loss: 0.1356 - binary
_accuracy: 0.9654 - val_loss: 0.2700 - val_binary_accuracy: 0.9597
Epoch 65/100
20/20 [==============================] - 6s 285ms/step - loss: 0.1310 - binary
_accuracy: 0.9665 - val_loss: 0.2315 - val_binary_accuracy: 0.9601
Epoch 66/100
20/20 [==============================] - 6s 290ms/step - loss: 0.1275 - binary
_accuracy: 0.9642 - val_loss: 0.3539 - val_binary_accuracy: 0.9617
Epoch 67/100
20/20 [==============================] - 6s 284ms/step - loss: 0.1368 - binary
_accuracy: 0.9657 - val_loss: 0.2354 - val_binary_accuracy: 0.9556
Epoch 68/100
20/20 [==============================] - 6s 276ms/step - loss: 0.1339 - binary
_accuracy: 0.9651 - val_loss: 0.2418 - val_binary_accuracy: 0.9591
Epoch 69/100
20/20 [==============================] - 5s 274ms/step - loss: 0.1389 - binary
_accuracy: 0.9636 - val_loss: 0.2677 - val_binary_accuracy: 0.9619
Epoch 70/100
20/20 [==============================] - 6s 276ms/step - loss: 0.1336 - binary
_accuracy: 0.9681 - val_loss: 0.2350 - val_binary_accuracy: 0.9605
Epoch 71/100
20/20 [==============================] - 6s 283ms/step - loss: 0.1294 - binary
_accuracy: 0.9644 - val_loss: 0.2670 - val_binary_accuracy: 0.9605
Epoch 72/100
20/20 [==============================] - 6s 289ms/step - loss: 0.1293 - binary
_accuracy: 0.9641 - val_loss: 0.2212 - val_binary_accuracy: 0.9484
Epoch 73/100
20/20 [==============================] - 6s 284ms/step - loss: 0.1344 - binary
_accuracy: 0.9660 - val_loss: 0.2962 - val_binary_accuracy: 0.9591
```

```
Epoch 74/100
20/20 [==============================] - 6s 289ms/step - loss: 0.1318 - binary
_accuracy: 0.9648 - val_loss: 0.2709 - val_binary_accuracy: 0.9568
Epoch 75/100
20/20 [==============================] - 6s 282ms/step - loss: 0.1263 - binary
_accuracy: 0.9661 - val_loss: 0.2823 - val_binary_accuracy: 0.9603
Epoch 76/100
20/20 [==============================] - 5s 269ms/step - loss: 0.1312 - binary
_accuracy: 0.9658 - val_loss: 0.3175 - val_binary_accuracy: 0.9597
Epoch 77/100
20/20 [==============================] - 6s 286ms/step - loss: 0.1284 - binary
_accuracy: 0.9646 - val_loss: 0.2678 - val_binary_accuracy: 0.9597
Epoch 78/100
20/20 [==============================] - 6s 292ms/step - loss: 0.1231 - binary
_accuracy: 0.9672 - val_loss: 0.3521 - val_binary_accuracy: 0.9593
Epoch 79/100
20/20 [==============================] - 6s 280ms/step - loss: 0.1363 - binary
_accuracy: 0.9634 - val_loss: 0.2637 - val_binary_accuracy: 0.9580
Epoch 80/100
20/20 [==============================] - 6s 280ms/step - loss: 0.1269 - binary
_accuracy: 0.9663 - val_loss: 0.2145 - val_binary_accuracy: 0.9580
Epoch 81/100
20/20 [==============================] - 6s 283ms/step - loss: 0.1372 - binary
_accuracy: 0.9643 - val_loss: 0.2175 - val_binary_accuracy: 0.9552
Epoch 82/100
20/20 [==============================] - 6s 283ms/step - loss: 0.1496 - binary
_accuracy: 0.9620 - val_loss: 0.2659 - val_binary_accuracy: 0.9615
Epoch 83/100
20/20 [==============================] - 6s 287ms/step - loss: 0.1390 - binary
_accuracy: 0.9635 - val_loss: 0.2626 - val_binary_accuracy: 0.9609
Epoch 84/100
20/20 [==============================] - 6s 283ms/step - loss: 0.1358 - binary
_accuracy: 0.9626 - val_loss: 0.3076 - val_binary_accuracy: 0.9585
Epoch 85/100
20/20 [==============================] - 6s 275ms/step - loss: 0.1237 - binary
_accuracy: 0.9682 - val_loss: 0.2799 - val_binary_accuracy: 0.9582
Epoch 86/100
20/20 [==============================] - 6s 290ms/step - loss: 0.1286 - binary
_accuracy: 0.9638 - val_loss: 0.2567 - val_binary_accuracy: 0.9583
Epoch 87/100
20/20 [==============================] - 6s 278ms/step - loss: 0.1225 - binary
_accuracy: 0.9680 - val_loss: 0.3209 - val_binary_accuracy: 0.9621
Epoch 88/100
20/20 [==============================] - 6s 285ms/step - loss: 0.1343 - binary
_accuracy: 0.9629 - val_loss: 0.3101 - val_binary_accuracy: 0.9623
Epoch 89/100
20/20 [==============================] - 6s 291ms/step - loss: 0.1499 - binary
_accuracy: 0.9641 - val_loss: 0.3058 - val_binary_accuracy: 0.9642
Epoch 90/100
20/20 [==============================] - 6s 284ms/step - loss: 0.1378 - binary
_accuracy: 0.9662 - val_loss: 0.2780 - val_binary_accuracy: 0.9634
Epoch 91/100
20/20 [==============================] - 6s 276ms/step - loss: 0.1318 - binary
_accuracy: 0.9653 - val_loss: 0.2524 - val_binary_accuracy: 0.9638
Epoch 92/100
20/20 [==============================] - 6s 289ms/step - loss: 0.1302 - binary
_accuracy: 0.9641 - val_loss: 0.2308 - val_binary_accuracy: 0.9642
Epoch 93/100
20/20 [==============================] - 6s 276ms/step - loss: 0.1275 - binary
_accuracy: 0.9658 - val_loss: 0.2558 - val_binary_accuracy: 0.9619
Epoch 94/100
20/20 [==============================] - 6s 292ms/step - loss: 0.1262 - binary
_accuracy: 0.9644 - val_loss: 0.2179 - val_binary_accuracy: 0.9583
Epoch 95/100
```

```
20/20 [==============================] - 6s 282ms/step - loss: 0.1296 - binary
_accuracy: 0.9663 - val_loss: 0.2503 - val_binary_accuracy: 0.9576
Epoch 96/100
20/20 [==============================] - 6s 284ms/step - loss: 0.1244 - binary
_accuracy: 0.9669 - val_loss: 0.2803 - val_binary_accuracy: 0.9611
Epoch 97/100
20/20 [==============================] - 6s 279ms/step - loss: 0.1182 - binary
_accuracy: 0.9688 - val_loss: 0.2641 - val_binary_accuracy: 0.9636
Epoch 98/100
20/20 [==============================] - 5s 273ms/step - loss: 0.1245 - binary
_accuracy: 0.9645 - val_loss: 0.3404 - val_binary_accuracy: 0.9619
Epoch 99/100
20/20 [==============================] - 6s 290ms/step - loss: 0.1261 - binary
_accuracy: 0.9649 - val_loss: 0.2542 - val_binary_accuracy: 0.9603
Epoch 100/100
20/20 [==============================] - 6s 286ms/step - loss: 0.1303 - binary
_accuracy: 0.9640 - val_loss: 0.2807 - val_binary_accuracy: 0.9638
```

In [50]:

```
predictions = model.predict(x_test)
print(predictions)
```

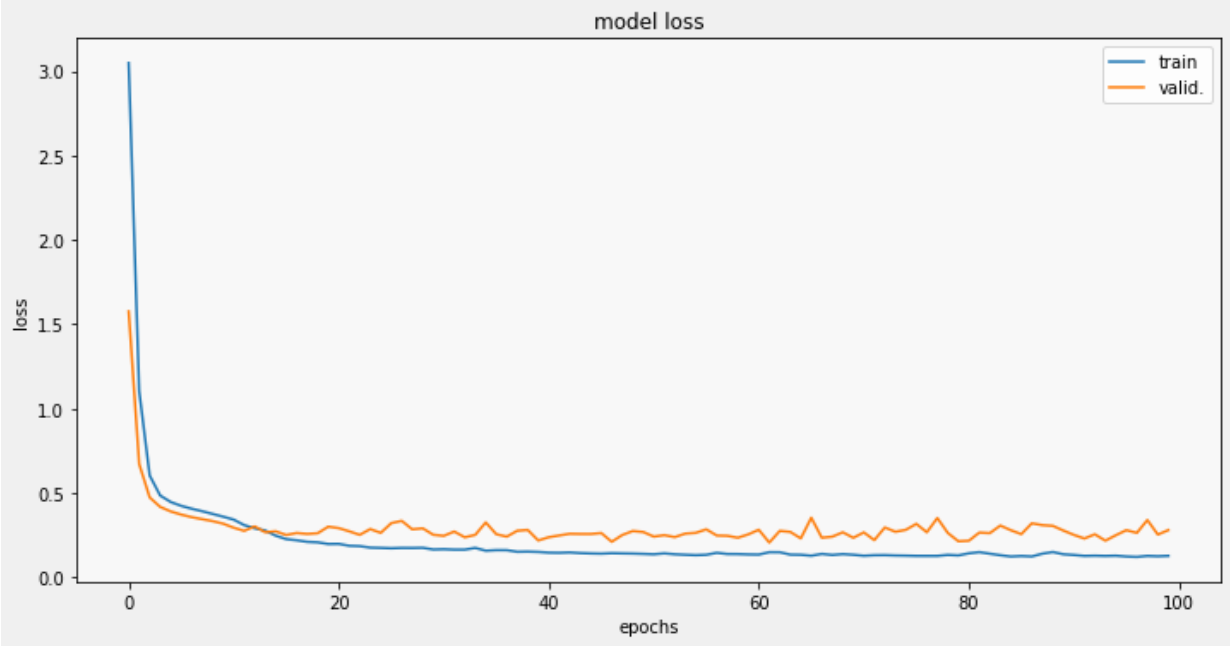
```
[[7.0840586e-05]
 [1.4306902e-06]
 [1.1695610e-06]
 ...
 [1.3989152e-06]
 [8.9077048e-07]
 [6.8215045e-06]]
```

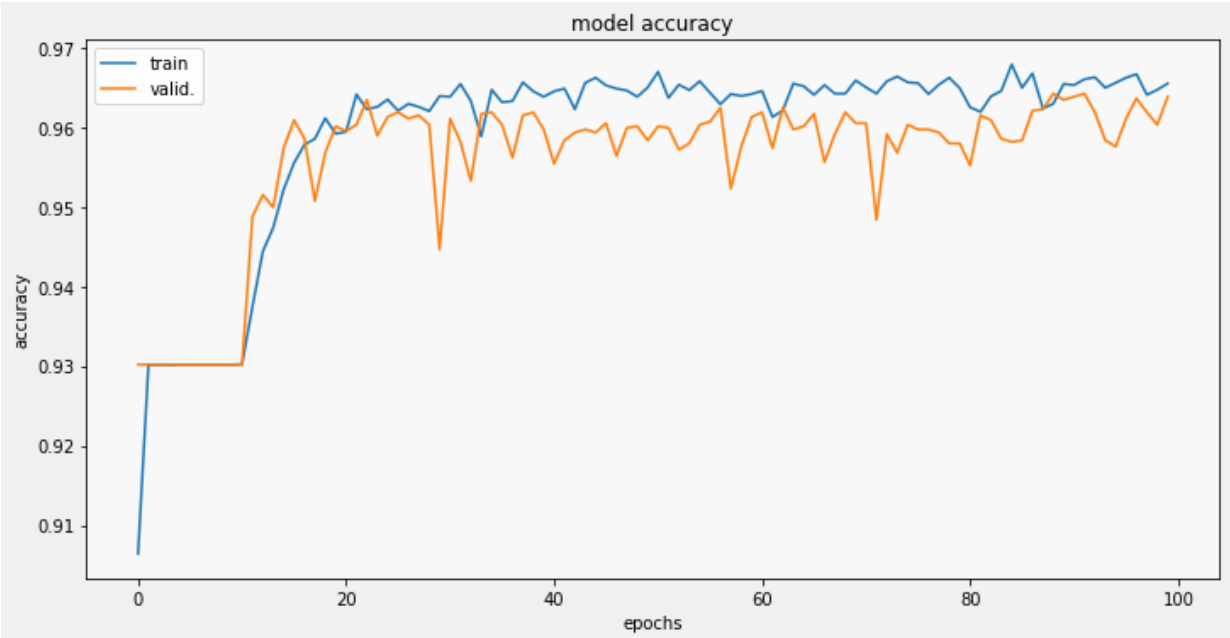
In [61]:

```
def display_training_curves(training, validation, title, subplot):
    _, ax = plt.subplots(figsize=(10,5), facecolor='#F0F0F0')
    plt.tight_layout()
    ax.set_facecolor('#F8F8F8')
    ax.plot(training)
    ax.plot(validation)
    ax.set_title('model '+ title)
    ax.set_ylabel(title)
    #ax.set_ylim(0.28,1.05)
    ax.set_xlabel('epochs')
    ax.legend(['train', 'valid.'])
```

In [62]:

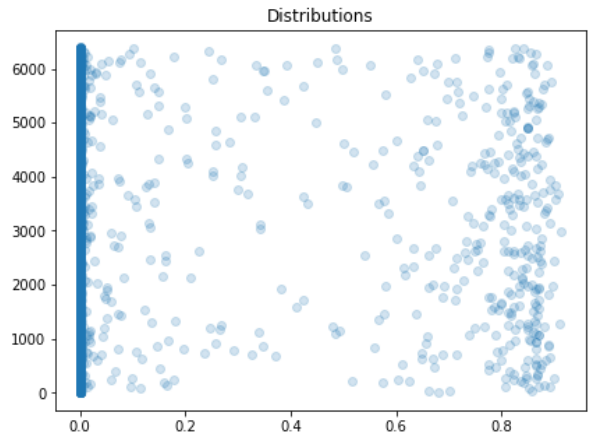
```
display_training_curves(
    history.history['loss'],
    history.history['val_loss'],
    'loss', 211)
display_training_curves(
    history.history['binary_accuracy'],
    history.history['val_binary_accuracy'],
    'accuracy', 212)
```

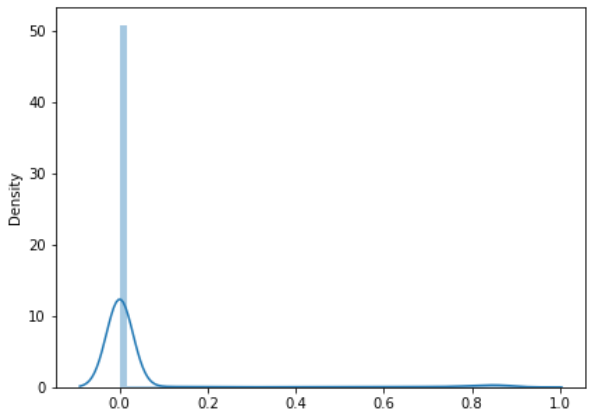




In [66]:

```
_, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,5))
ax1.scatter(predictions, range(0, len(predictions)), alpha=0.2)
ax1.set_title("Distributions")
ax2 = sns.distplot(predictions)
```





In [68]:

```
final_test_df = pd.read_csv('test_tweets.csv')
ftest = final_test_df.copy()
```

```
ftest.drop(columns=['id'], axis=1, inplace=True)

ftest['tweet'] = ftest['tweet'].apply(remove_emoji)
ftest['tweet'] = ftest['tweet'].apply(clean_text)

f_test = np.array(tokenizer.texts_to_sequences(ftest['tweet'].tolist()))
f_test = pad_sequences(f_test, padding='post', maxlen=maxlen)

display(f_test)
```

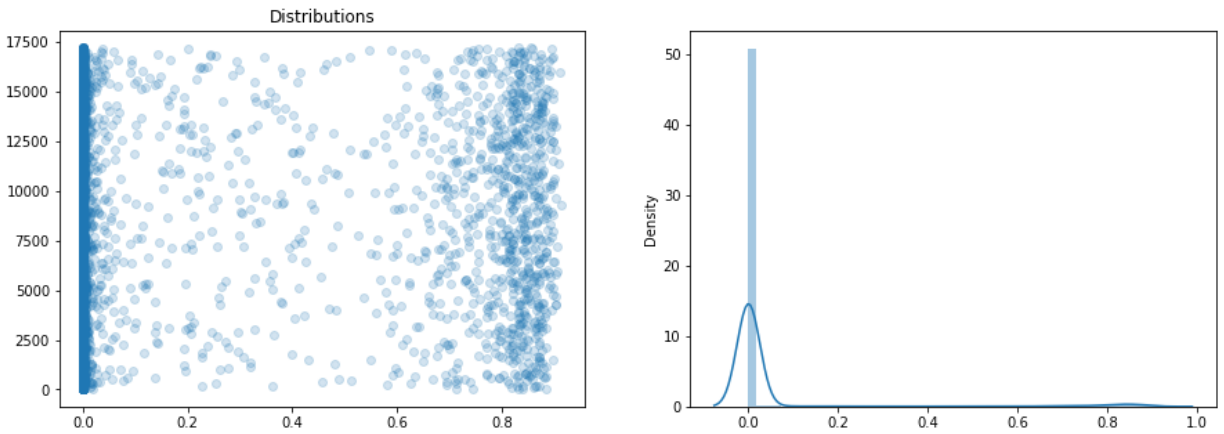
```
array([[25732,     1,  9182, ...,     0,     0,     0],
       [    2,   185, 20821, ...,     0,     0,     0],
       [  458,   718,   602, ...,     0,     0,     0],
       ...,
       [  580,     1,    18, ...,     0,     0,     0],
       [    7,    36,   318, ...,     0,     0,     0],
       [  296,   565,   127, ...,     0,     0,     0]], dtype=int32)
```

In [69]:

```
predictions_f_test = model.predict(f_test)
```

In [71]:

```
_, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,5))
ax1.scatter(predictions_f_test, ftest.index, alpha=0.2)
ax1.set_title("Distributions")
ax2 = sns.distplot(predictions_f_test)
```



In [73]:

```
!pip3 freeze > requirements.txt
```